TODAY’S TOPIC:

- Designing classes
- Chapter 5: Class design guidelines
MODELING THE REAL WORLD

• A goal of OO Design is to model the real world.

  – This allows the systems that we build to be similar to how people actually think about the system.
    • A top-down procedural approach, requires a designer to think of the data and procedures separately.
    • The OO Design similar to how we design other systems. (Buildings, cars, and other types of engineering).

OBJECTS WORKING TOGETHER

• A class will interact with other classes.
  • Sometimes it will provide a service for another object.
  • Sometimes it will request a service from another object.

• Objects should be designed with reuse in mind.
  • A goal is to make a class as flexible as possible,
  • You don’t want to provide unnecessary functionality, but the design should allow for future enhancements.
How to decide what classes to create?

- A common mistake of a novice OO developer is to create too few classes.

- The ones they do create tend to be large and contain many different procedures and instance variables.

- The class methods themselves tend to be long and complex.

A good OO Design will contain:

- Many little specialized classes, each containing a cohesive set of data and methods.

- Class methods will be short (broken down into sub methods).
HOW TO IDENTIFY CLASSES

• A class represents a single concept or object.

• The components of a class should be cohesive and all the data and methods fit together with each other.

• This is another way of saying, not to make a large class that does it all.

SIMPLE CLASS METHODS

• Methods should be simple enough to write with a small amount of code.

• Some OO developers will even give a limit, say 15 or 20 lines.

• If you are writing a method and it seems too long or complex to follow, break it apart into multiple methods.
MINIMIZE SIDE AFFECTS

- A side effect of a method is a externally observable change in the data state.

- All mutator (setter) methods will change an instance variable as a side effect.

- Sometimes other will as well, recall the grow() method in our rectangle class.

LET THE USER INTERFACE CLASSES DO I/O

- This example mutates the System.out object.

```java
    public void printBalance() // Not recommended
    {
        System.out.println("The balance is now "+balance);
    }
```

- This is also a side effect to be avoided.

- The I/O and user interface should be done by presentation objects (recall the n-tier design).